**集美大学计算机工程学院实验报告**

|  |  |  |
| --- | --- | --- |
| **课程名称**：Java程序设计 | **班级**：软件1912 | **实验成绩**： |
| **指导教师**：李　旺 | **姓名**：游嘉升 |  |
| **实验项目名称**：接口与内部类 | **学号**：201921122050 | **上机实践日期**：10月30日 |
| **实验项目编号**： | **组号**： | **上机实践时间**： 2 学时 |

**一、目的**

掌握接口的定义与实现；熟悉内部类机制；能够使用Comparable、Comparator接口；会使用简单的Lambda表达式。

**二、实验内容与设计思想**

1. Shape与Comparable接口

改写实验4中的抽象类Shape，让其实现Comparable接口，实现对面积的排序；在测试类中创建Shape类型的数组或动态数组，里面有若干个Circle、Rectangle和Triangle对象，然后利用Arrays.sort或Collections.sort对该Shape数组或动态数组进行排序；输出排序后Shapes数组。

**注意：所有的代码都应该放在合适的包中（包名自定）。**

2. Staff与Comparator接口

(1)针对实验4编写的Staff类，分别编写NameComparator，AgeComparator和ComplexComparator比较器。其中ComplexComparator为复合比较器，至少选用两个关键字来排序，关键字选择、次序及升降序均由同学们自行设计。

(2)编写一个测试类，在测试类中添加若干个Staff, Teacher, SecurityGuard, Dean实例(个数及内容自定)，并在测试类中定义并测试如下方法：①编写一个方法private static void printName(Staff[] staffs)打印出每个人的名字；②编写一个方法private static void printSalary(Staff[] staffs)打印出Staff类或者其子类对象的薪水(注意：Staff的薪水只有salary，Teacher的薪水为salary+postAllowance，SecurityGuard的薪水为salary+dangerousAllowance，而Dean的薪水则为salary+postAllowance+adminAward)；③编写一方法private static void sortBySalary(Staff[] staffs)，支持对Staff类及其子类按照各自的薪水降序排序；④编写一方法private static void sortByAge(Staff[] staffs)，对Staff对象按照年龄升序排序，再编写一个方法按name升序进行排序；⑤编写一方法sortByDateHired，支持对Staff类及其子类按照各自的dateHired升序排序，可以使用java.util.Date或java.time.LocalDateTime类型的compareTo方法。

3. Lambda表达式

使用Lambda表达式简化上述“Staff与Comparator接口”题目中的NameComparator，AgeComparator和ComplexComparator比较器。测试方法与上述相同。

4. 枚举类型

编写枚举类型City，表示城市，定义若干常量(如厦门、北京、上海等)，每个City常量都包含一个中文的描述信息(即需要一个带一个参数的构造方法)。

编写枚举类型University，表示大学，定义若干常量(如集美大学、厦门大学、清华大学等，允许自定义，不限)，每个University常量包含两个信息：中文描述及所在的城市(City类型)。

编写简单的测试类，使用上述枚举类型赋值及输出。

**三、实验使用环境**

Java version "13.0.2"

IntelliJ IDEA 2020.2.2 x64

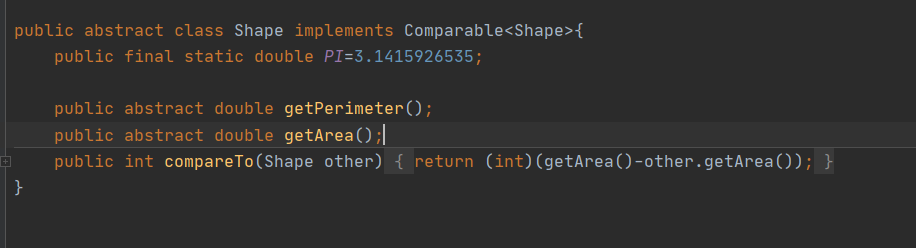
**四、实验步骤和调试过程**

1. Shape与Comparable接口（代码见附件shapeandcomparable）

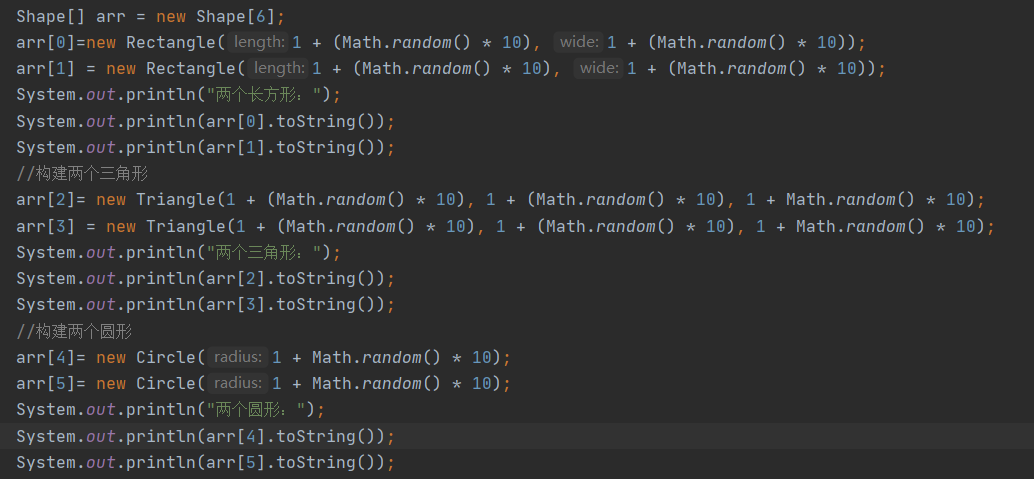
1）设计思想：定义Shape抽象类实现一个接口Comparable，并在Shape类中完成compareTo的重写。

2）关键代码：

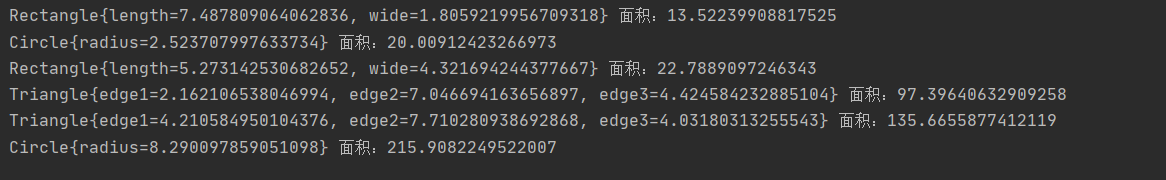
Comparable接口的定义以及compareTo的实现



3）测试数据：



4）测试结果：



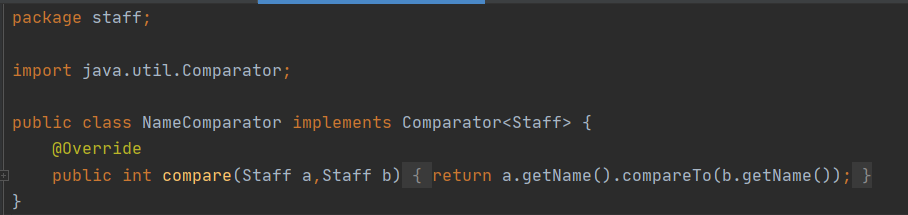
2. Staff与Comparator接口（代码见附件staffandcomparator）

1. 设计思想：

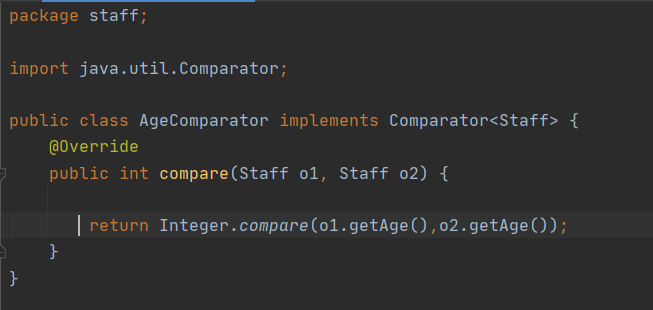
定义多个比较器实现Comparator接口，并在内部实现compare方法。

1. 关键代码：

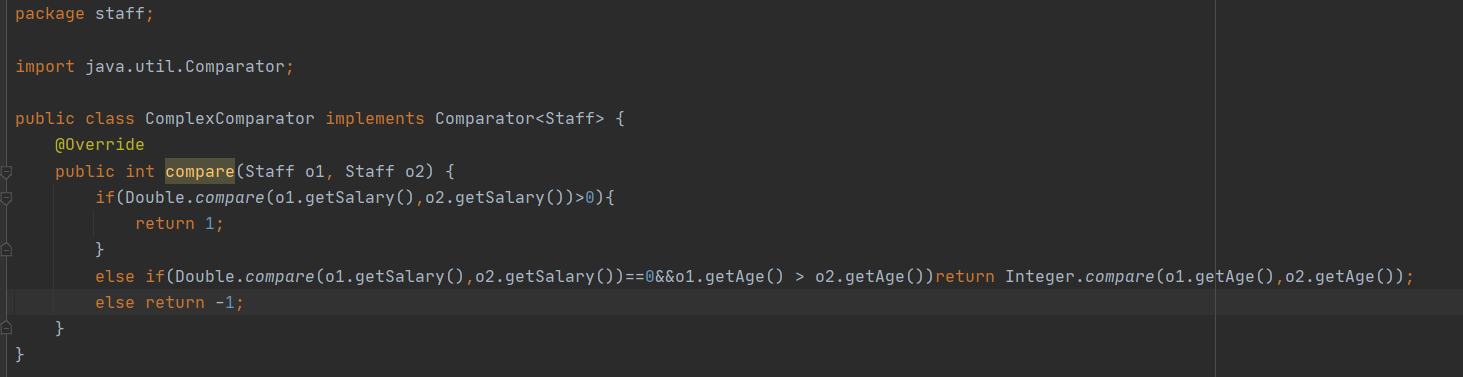
NameComparator的定义



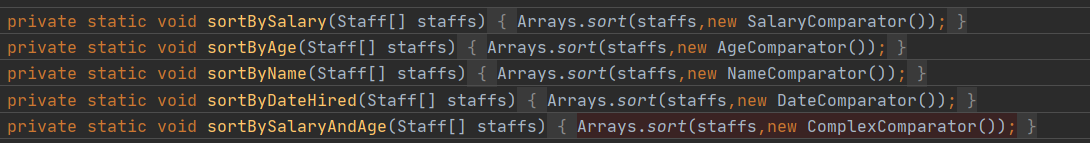
AgeComparator的定义



ComplexComparator的定义



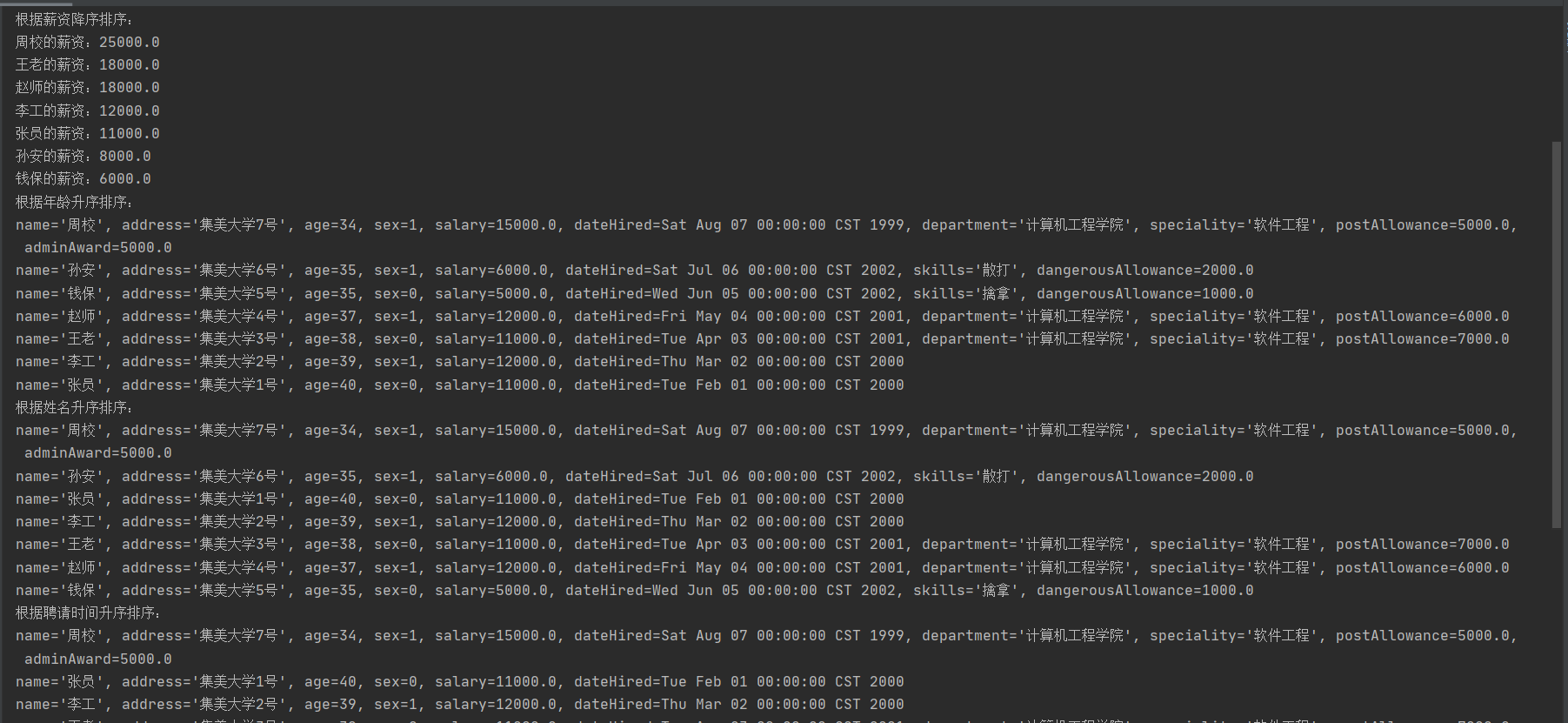
各种sort方法的定义



1. 测试数据：



1. 测试结果：



3. Lambda表达式（代码见附件lambda）

1）设计思想：

在多种sort方法中的Arrays.sort中使用Lambda表达式实现比较器的功能

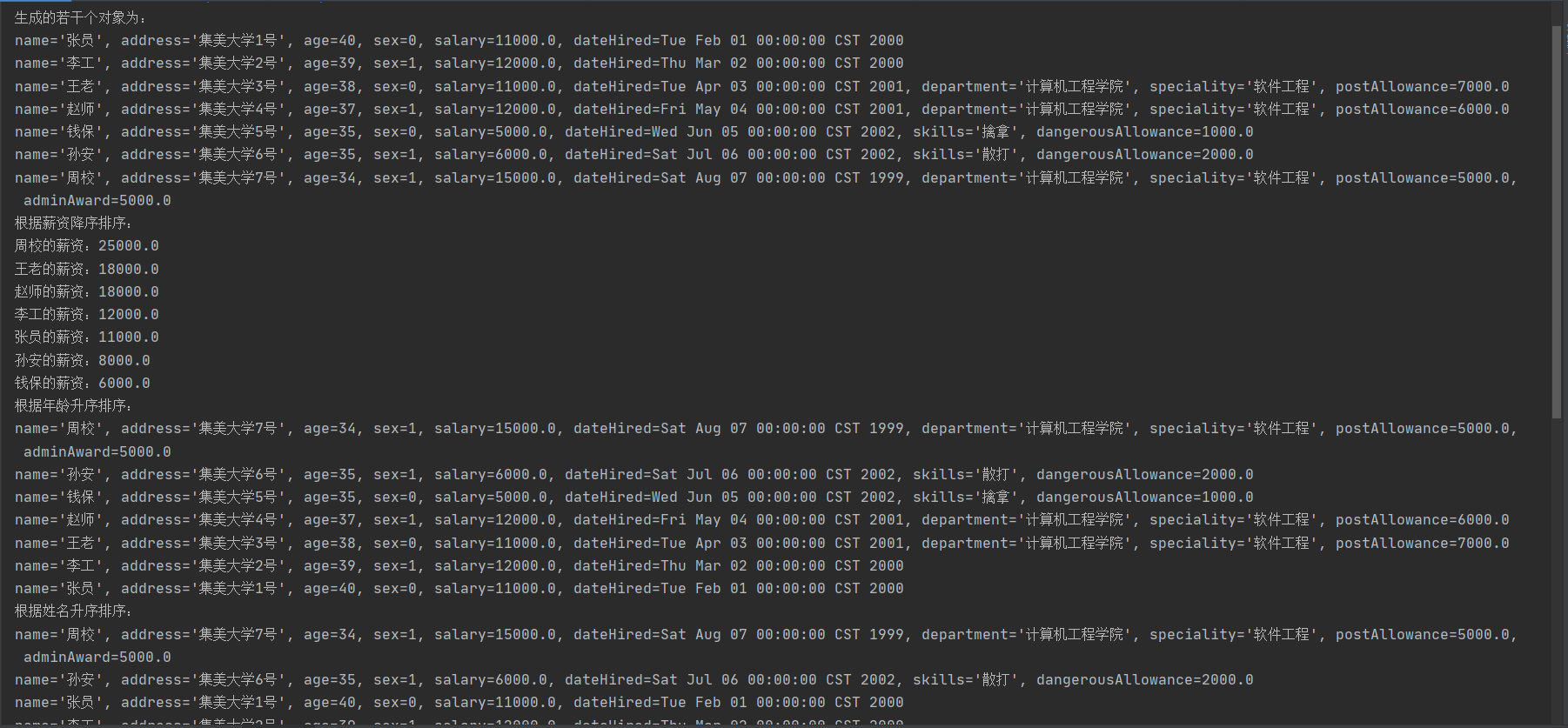
2）关键代码：

Arrays.sort中的Lambda



3）测试数据：测试数据同上

4）测试结果：



4. 枚举类型（代码见附件enums）

1）设计思想：定义City和University枚举类以及构造方法和getter方法

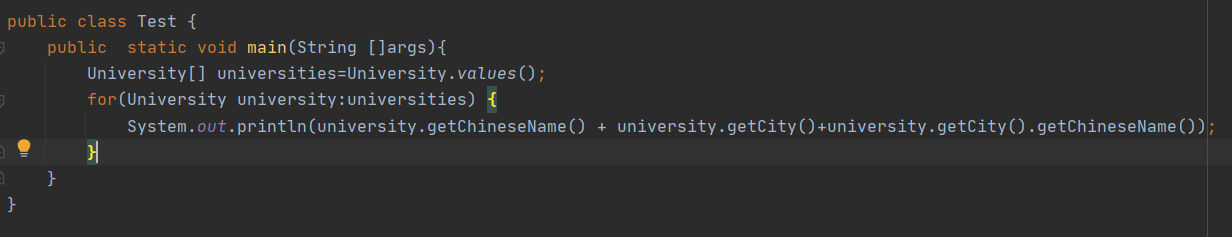
2）关键代码：

City定义

University定义



Test方法定义



3）测试数据：见关键代码

4）测试结果：



**五、实验小结**

1．实验中遇到的问题及解决过程

①比较器ComplexComparator排序错误

解决过程：发现一个关键字重写compare方法时调用类自带的compareTo方法返回值是1，0，-1，而自己在ComplexComparator比较器中重写compare函数返回值只有1和0，导致ComplexComparator排序错误。

2．实验中产生的错误及原因分析

Comparable和Comparator分不清，每次错误其实都是因为混淆了概念。

3．实验体会和收获

学会了使用Comparable和Comparator的正确使用方法，以后想要排序只需要重写比较器然后调用Arrays.sort或者Collection.sort，理解了Comparable和Comparator的区别，一个是类本身的内部定义，而一个是由外部提供的一种比较方法，还学习了枚举类，具体作用还不清楚，之后应该会用到。

**六、附录**

[1][美] 凯.S.霍斯特曼（Cay S. Horstmann）.Java核心技术·卷 I（原书第10版）[M].机械工业出版社